Pandas

November 30, 2024

1 Pandas

Two data structure

- Series(1D)
- DataFrame(2D)

Pandas holds precisely one data type components of the DataFrame

- Index
- Columns
- Data

A DataFrame has two axes—

- a vertical axis (the index) and
- a horizontalaxis(the columns)

Pandas borrows convention from NumPy and uses the integers 0/1 as another way of referring to the vertical/horizontal axis

Pandas uses NaN (not a number) to represent missing values.

2 Series

A Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating-point numbers, Python objects, and more). It consists of two main components: the data and the index. The data can be provided using a NumPy array, a Python list, or a scalar value.

Index: It is a sequence of labels which identifies each element in the Series. By default, the index starts from 0 and increments by 1, but you can customize it.Index: It is a sequence of labels which identifies each element in the Series.

[114]:
```
import pandas as pd
```

[115]:
```
movie = pd.read_csv('movie.csv')
movie.head()
```

[115]:
```
   color      director_name  num_critic_for_reviews  duration  \
0  Color      James Cameron                   723.0     178.0
1  Color     Gore Verbinski                   302.0     169.0
2  Color         Sam Mendes                   602.0     148.0
3  Color  Christopher Nolan                   813.0     164.0
4    NaN        Doug Walker                     NaN       NaN

   director_facebook_likes  actor_3_facebook_likes      actor_2_name  \
0                      0.0                   855.0  Joel David Moore
1                    563.0                  1000.0     Orlando Bloom
2                      0.0                   161.0      Rory Kinnear
3                  22000.0                 23000.0    Christian Bale
4                    131.0                     NaN        Rob Walker

   actor_1_facebook_likes        gross                           genres  …  \
0                  1000.0  760505847.0  Action|Adventure|Fantasy|Sci-Fi  …
1                 40000.0  309404152.0         Action|Adventure|Fantasy  …
2                 11000.0  200074175.0        Action|Adventure|Thriller  …
3                 27000.0  448130642.0                  Action|Thriller  …
4                   131.0          NaN                     Documentary  …

  num_user_for_reviews language  country  content_rating       budget  \
0               3054.0  English      USA           PG-13  237000000.0
1               1238.0  English      USA           PG-13  300000000.0
2                994.0  English       UK           PG-13  245000000.0
3               2701.0  English      USA           PG-13  250000000.0
4                  NaN      NaN      NaN             NaN          NaN

   title_year actor_2_facebook_likes imdb_score  aspect_ratio  \
0      2009.0                 936.0        7.9          1.78
1      2007.0                5000.0        7.1          2.35
2      2015.0                 393.0        6.8          2.35
3      2012.0               23000.0        8.5          2.35
4         NaN                  12.0        7.1           NaN

  movie_facebook_likes
0                33000
1                    0
2                85000
3               164000
4                    0

[5 rows x 28 columns]
```

[116]:
```
index= movie.index
columns = movie.columns
data = movie.values
```

[117]:
```
index
```

[117]:
```
RangeIndex(start=0, stop=4916, step=1)
```

[118]:
```
columns
```

[118]:
```
Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',
       'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
       'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
       'movie_title', 'num_voted_users', 'cast_total_facebook_likes',
       'actor_3_name', 'facenumber_in_poster', 'plot_keywords',
       'movie_imdb_link', 'num_user_for_reviews', 'language', 'country',
       'content_rating', 'budget', 'title_year', 'actor_2_facebook_likes',
       'imdb_score', 'aspect_ratio', 'movie_facebook_likes'],
      dtype='object')
```

[119]:
```
data
```

[119]:
```
array([['Color', 'James Cameron', 723.0, …, 7.9, 1.78, 33000],
       ['Color', 'Gore Verbinski', 302.0, …, 7.1, 2.35, 0],
       ['Color', 'Sam Mendes', 602.0, …, 6.8, 2.35, 85000],
       …,
       ['Color', 'Benjamin Roberds', 13.0, …, 6.3, nan, 16],
       ['Color', 'Daniel Hsia', 14.0, …, 6.3, 2.35, 660],
       ['Color', 'Jon Gunn', 43.0, …, 6.6, 1.85, 456]], dtype=object)
```

[120]:
```
type(index)
```

[120]:
```
pandas.core.indexes.range.RangeIndex
```

[121]:
```
type(columns)
```

[121]:
```
pandas.core.indexes.base.Index
```

[122]:
```
type(data)
```

[122]:
```
numpy.ndarray
```

[123]:
```
issubclass(pd.RangeIndex, pd.Index)
```

[123]:
```
True
```

[124]:
```
index.values
```

[124]:
```
array([   0,    1,    2, …, 4913, 4914, 4915])
```

[125]:
```
movie.dtypes # dtypes attribute to display each column along with its data type:
```

[125]:
```
color                       object
director_name               object
num_critic_for_reviews     float64
duration                   float64
director_facebook_likes    float64
actor_3_facebook_likes     float64
actor_2_name                object
actor_1_facebook_likes     float64
gross                      float64
genres                      object
actor_1_name                object
movie_title                 object
num_voted_users              int64
cast_total_facebook_likes    int64
actor_3_name                object
facenumber_in_poster       float64
plot_keywords               object
movie_imdb_link             object
num_user_for_reviews       float64
language                    object
country                     object
content_rating              object
budget                     float64
title_year                 float64
actor_2_facebook_likes     float64
imdb_score                 float64
aspect_ratio               float64
movie_facebook_likes         int64
dtype: object
```

[126]:
```
"""
Example We will start with a basic example using a Python list. Suppose
you have a list of weekly temperatures: Pandas offers a data structure
called a which is ideal for storing and working with this type of data.
"""
Temp = [24, 24, 34, 56, 66, 76]
series = pd.Series(Temp)
print(series)
```

```
0    24
1    24
2    34
3    56
4    66
5    76
dtype: int64
```

[127]:
```
"""
Example In this example, we are using a scalar value. Suppose you want
to create a Series with the same value repeated multiple times. Let's say
you want a Series with the value 10 repeated 5 times
"""
Value = 10
series = pd.Series(Value, index=[0,1,2,3,4])
print(series)
```

```
0    10
1    10
2    10
3    10
4    10
dtype: int64
```

[128]:
```
"""
Example Using default index
Let's consider the previous example of the temperature Series. The default
index labels are assigned automatically when we create the Series.
"""
Temperatures = [25, 28, 30, 26, 29, 31, 27]
series = pd.Series(Temperatures)
print(series)
```

```
0    25
1    28
2    30
3    26
4    29
5    31
6    27
dtype: int64
```

[129]:
```
"""Example Using custom index
Suppose you have a Series representing the ages of different people, and
you want to assign custom labels to each age.
"""
"""
In this example, we assigned custom index labels (names) to each age in
the Series, making it easier to identify the age of each person
"""

Ages = [25, 30, 35, 28, 32]
index_labels = ['John', 'Jane', 'Mike', 'Emily', 'Alex']
series = pd.Series(Ages, index=index_labels)
print(series)
```

```
John     25
Jane     30
Mike     35
Emily    28
Alex     32
dtype: int64
```

3 DataFrame

A DataFrame in Pandas is a two-dimensional labeled data structure that can hold multiple columns. It can be thought of as a table or spreadsheet where each column represents a variable or attribute, and each row represents a specific observation or record.

A DataFrame consists of three main components: data, index, and columns.

The data component of a DataFrame represents the actual values in the table. It can be created from various data structures, such as Python dictionaries, NumPy arrays, or other DataFrames

[130]:
```
#Example Creating a DataFrame from a Python dictionary:
data = {'Name': ['John', 'Jane', 'Mike'],
'Age': [25, 30, 35],
'City=': ['New York', 'Paris', 'London']}
df = pd.DataFrame(data)
print(df)
```

```
   Name  Age     City=
0  John   25  New York
1  Jane   30     Paris
2  Mike   35    London
```

The index component of a DataFrame represents the labels assigned to each row. It helps to uniquely identify and access specific rows in the DataFrame. By default, Pandas assigns a numeric index starting from 0, but you can customize it with your own labels.

[131]:
```
#Example Customizing the index labels of a DataFrame:
data = {'Name': ['John', 'Jane', 'Mike'],
'Age': [25, 30, 35],
'City': ['New York', 'Paris', 'London']}
df = pd.DataFrame(data, index=['A', 'B', 'C'])
print(df)
```

```
   Name  Age      City
A  John   25  New York
B  Jane   30     Paris
C  Mike   35    London
```

4 Defining Datatypes

Python has several built-in datatypes, such as and so on. However, pandas borrows its datatypes from another Python library called NumPy, which is a library for scientific computing. NumPy has more datatypes than Python, such as and so on. These datatypes allow us to specify the size and precision of the data.

Pandas also has some datatypes that are specific to pandas, such as and category. These datatypes allow us to work with dates and times and categorical data.

Pandas will automatically assign a suitable datatype to each column or Series based on the values in it. We can also specify our own datatype by using the dtype argument in the constructor.

The object datatype is used to store any type of data that is not numeric or boolean. It can store strings, mixed types or Python objects. The object datatype is also used when pandas cannot infer a specific datatype for a column or

For example: #Create a Series of strings

[132]:
```
s = pd.Series(['apple', 'banana', 'cherry'])
```

[133]:
```
#Check the datatype of the Series
```

[134]:
```
print(s.dtype)
```

```
object
```

[135]:
```
""" We can also create a DataFrame with object columns by using a dictionary of
lists or Series. For example:
"""
# Create a DataFrame with object columns
df = pd.DataFrame({'name': ['Alice', 'Bob', 'Charlie'],
'gender': ['F', 'M', 'M'],
'hobby': ['reading', 'gaming', 'cooking']})
# Check the datatypes of all the columns
print(df.dtypes)
```

```
name      object
gender    object
hobby     object
dtype: object
```

5 Loading Data from Files and the Web for Pandas

Pandas supports CSV, Excel, JSON, HTML, and SQL files

5.1 Loading Data from CSV Files Using pandas.read_csv()

[136]:
```
# Read data from a CSV file
df = pd.read_csv('movie.csv')
```

[137]:
```
df.head()
```

[137]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres	…	num_user_for_reviews	language	country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	…	3054.0	English	USA	PG-13	237000000.0	2009.0	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action|Adventure|Fantasy	…	1238.0	English	USA	PG-13	300000000.0	2007.0	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action|Adventure|Thriller	…	994.0	English	UK	PG-13	245000000.0	2015.0	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action|Thriller	…	2701.0	English	USA	PG-13	250000000.0	2012.0	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	Documentary	…	NaN	NaN	NaN	NaN	NaN	NaN	12.0	7.1	NaN	

```
4                    0

[5 rows x 28 columns]
```

[138]:
```
"""If we want to use a different row as the column names, we can pass the row
number to this parameter. For example, if we want to use the second row as
the column names, we can pass
"""
df = pd.read_csv('movie.csv', header =2)
```

[139]:
```
df.head()
```

[139]:
```
   Color     Gore Verbinski  302.0  169.0    563.0   1000.0    Orlando Bloom  \
0  Color          Sam Mendes  602.0  148.0      0.0    161.0     Rory Kinnear
1  Color  Christopher Nolan  813.0  164.0  22000.0  23000.0   Christian Bale
2    NaN        Doug Walker    NaN    NaN    131.0      NaN       Rob Walker
3  Color     Andrew Stanton  462.0  132.0    475.0    530.0  Samantha Morton
4  Color          Sam Raimi  392.0  156.0      0.0   4000.0     James Franco

   40000.0  309404152.0   Action|Adventure|Fantasy  …  1238.0  English  USA  \
0  11000.0  200074175.0  Action|Adventure|Thriller  …   994.0  English   UK
1  27000.0  448130642.0            Action|Thriller  …  2701.0  English  USA
2    131.0          NaN                Documentary  …     NaN      NaN  NaN
3    640.0   73058679.0    Action|Adventure|Sci-Fi  …   738.0  English  USA
4  24000.0  336530303.0   Action|Adventure|Romance  …  1902.0  English  USA

   PG-13  300000000.0  2007.0   5000.0  7.1  2.35       0
0  PG-13  245000000.0  2015.0    393.0  6.8  2.35   85000
1  PG-13  250000000.0  2012.0  23000.0  8.5  2.35  164000
2    NaN          NaN     NaN     12.0  7.1   NaN       0
3  PG-13  263700000.0  2012.0    632.0  6.6  2.35   24000
4  PG-13  258000000.0  2007.0  11000.0  6.2  2.35       0

[5 rows x 28 columns]
```

[140]:
```
df = pd.read_csv('movie.csv', usecols=["color", "director_name"])
df
```

[140]:
```
      color      director_name
0     Color      James Cameron
1     Color     Gore Verbinski
2     Color         Sam Mendes
3     Color  Christopher Nolan
4       NaN        Doug Walker
…     …                  …
4911  Color        Scott Smith
4912  Color                NaN
```

```
4913  Color   Benjamin Roberds
4914  Color        Daniel Hsia
4915  Color           Jon Gunn

[4916 rows x 2 columns]
```

[141]:
```
usecols=["Color", "Orlando Bloom"]
```

[142]:
```
df.head()
```

[142]:
```
   color      director_name
0  Color      James Cameron
1  Color     Gore Verbinski
2  Color         Sam Mendes
3  Color  Christopher Nolan
4    NaN        Doug Walker
```

[143]:
```
df = pd.read_csv('movie.csv')
df
```

[143]:
```
      color      director_name  num_critic_for_reviews  duration  \
0     Color      James Cameron                   723.0     178.0
1     Color     Gore Verbinski                   302.0     169.0
2     Color         Sam Mendes                   602.0     148.0
3     Color  Christopher Nolan                   813.0     164.0
4       NaN        Doug Walker                     NaN       NaN
…     …                  …                     …       …
4911  Color        Scott Smith                     1.0      87.0
4912  Color                NaN                    43.0      43.0
4913  Color   Benjamin Roberds                    13.0      76.0
4914  Color        Daniel Hsia                    14.0     100.0
4915  Color           Jon Gunn                    43.0      90.0

      director_facebook_likes  actor_3_facebook_likes      actor_2_name  \
0                         0.0                   855.0  Joel David Moore
1                       563.0                  1000.0     Orlando Bloom
2                         0.0                   161.0      Rory Kinnear
3                     22000.0                 23000.0    Christian Bale
4                       131.0                     NaN        Rob Walker
…                       …                     …              …
4911                      2.0                   318.0     Daphne Zuniga
4912                      NaN                   319.0     Valorie Curry
4913                      0.0                     0.0     Maxwell Moody
4914                      0.0                   489.0     Daniel Henney
4915                     16.0                    16.0  Brian Herzlinger

      actor_1_facebook_likes        gross                         genres  \
```

```
0                    1000.0  760505847.0  Action|Adventure|Fantasy|Sci-Fi
1                   40000.0  309404152.0         Action|Adventure|Fantasy
2                   11000.0  200074175.0        Action|Adventure|Thriller
3                   27000.0  448130642.0                  Action|Thriller
4                     131.0          NaN                     Documentary
…                         …            …                               …
4911                  637.0          NaN                    Comedy|Drama
4912                  841.0          NaN     Crime|Drama|Mystery|Thriller
4913                    0.0          NaN            Drama|Horror|Thriller
4914                  946.0      10443.0             Comedy|Drama|Romance
4915                   86.0      85222.0                     Documentary

      … num_user_for_reviews language  country  content_rating       budget  \
0     …               3054.0  English      USA           PG-13  237000000.0
1     …               1238.0  English      USA           PG-13  300000000.0
2     …                994.0  English       UK           PG-13  245000000.0
3     …               2701.0  English      USA           PG-13  250000000.0
4     …                  NaN      NaN      NaN             NaN          NaN
…   …                    …        …        …               …            …
4911  …                  6.0  English   Canada             NaN          NaN
4912  …                359.0  English      USA           TV-14          NaN
4913  …                  3.0  English      USA             NaN       1400.0
4914  …                  9.0  English      USA           PG-13          NaN
4915  …                 84.0  English      USA              PG       1100.0

      title_year actor_2_facebook_likes imdb_score  aspect_ratio  \
0         2009.0                  936.0        7.9          1.78
1         2007.0                 5000.0        7.1          2.35
2         2015.0                  393.0        6.8          2.35
3         2012.0                23000.0        8.5          2.35
4            NaN                   12.0        7.1           NaN
…             …                      …          …             …
4911      2013.0                  470.0        7.7           NaN
4912         NaN                  593.0        7.5         16.00
4913      2013.0                    0.0        6.3           NaN
4914      2012.0                  719.0        6.3          2.35
4915      2004.0                   23.0        6.6          1.85

     movie_facebook_likes
0                   33000
1                       0
2                   85000
3                  164000
4                       0
…                       …
4911                   84
4912                32000
```

```
4913                   16
4914                  660
4915                  456

[4916 rows x 28 columns]
```

[144]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4916 entries, 0 to 4915
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   color                      4897 non-null   object
 1   director_name              4814 non-null   object
 2   num_critic_for_reviews     4867 non-null   float64
 3   duration                   4901 non-null   float64
 4   director_facebook_likes    4814 non-null   float64
 5   actor_3_facebook_likes     4893 non-null   float64
 6   actor_2_name               4903 non-null   object
 7   actor_1_facebook_likes     4909 non-null   float64
 8   gross                      4054 non-null   float64
 9   genres                     4916 non-null   object
 10  actor_1_name               4909 non-null   object
 11  movie_title                4916 non-null   object
 12  num_voted_users            4916 non-null   int64
 13  cast_total_facebook_likes  4916 non-null   int64
 14  actor_3_name               4893 non-null   object
 15  facenumber_in_poster       4903 non-null   float64
 16  plot_keywords              4764 non-null   object
 17  movie_imdb_link            4916 non-null   object
 18  num_user_for_reviews       4895 non-null   float64
 19  language                   4902 non-null   object
 20  country                    4911 non-null   object
 21  content_rating             4616 non-null   object
 22  budget                     4432 non-null   float64
 23  title_year                 4810 non-null   float64
 24  actor_2_facebook_likes     4903 non-null   float64
 25  imdb_score                 4916 non-null   float64
 26  aspect_ratio               4590 non-null   float64
 27  movie_facebook_likes       4916 non-null   int64
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

[153]:
```
df = pd.read_csv('movie.csv', dtype='category')
df[["title_year", "Type"]].dtypes
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[153], line 2
      1 df = pd.read_csv('movie.csv', dtype='category')
----> 2 df[["title_year", "Type"]].dtypes

File ~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/frame.py:
 ↪4108, in DataFrame.__getitem__(self, key)
   4106     if is_iterator(key):
   4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   4110 # take() does not accept boolean indexers
   4111 if getattr(indexer, "dtype", None) == bool:

File ~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/indexes/
 ↪base.py:6200, in Index._get_indexer_strict(self, key, axis_name)
   6197 else:
   6198     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
   6202 keyarr = self.take(indexer)
   6203 if isinstance(key, Index):
   6204     # GH 42790 - Preserve name from an Index

File ~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/indexes/
 ↪base.py:6252, in Index._raise_if_missing(self, key, indexer, axis_name)
   6249     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6251 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 6252 raise KeyError(f"{not_found} not in index")

KeyError: "['Type'] not in index"
```

6 Descriptive Analytics

The objective of descriptive analytics is simple comprehension of data using data summarization, basic statistical measures and visualization.

[154]:
```
import pandas as pd
df = pd.read_csv('IPL IMB381IPL2013.csv')
df
```

[154]:

	Sl.NO.	PLAYER NAME	AGE	COUNTRY	TEAM	PLAYING ROLE	T-RUNS	T-WKTS	\
0	1	Abdulla, YA	2	SA	KXIP	Allrounder	0	0	
1	2	Abdur Razzak	2	BAN	RCB	Bowler	214	18	
2	3	Agarkar, AB	2	IND	KKR	Bowler	571	58	
3	4	Ashwin, R	1	IND	CSK	Bowler	284	31	

```
4         5  Badrinath, S    2     IND    CSK      Batsman      63       0
..      …           …  …     …     …          …     …     …
125     126     Yadav, AS    2     IND     DC      Batsman       0       0
126     127   Younis Khan    2     PAK     RR      Batsman    6398       7
127     128  Yuvraj Singh    2     IND  KXIP+      Batsman    1775       9
128     129   Zaheer Khan    2     IND    MI+       Bowler    1114     288
129     130    Zoysa, DNT    2      SL     DC       Bowler     288      64

     ODI-RUNS-S  ODI-SR-B  …    SR-B  SIXERS  RUNS-C  WKTS  AVE-BL   ECON  \
0             0      0.00  …    0.00       0     307    15   20.47   8.90
1           657     71.41  …    0.00       0      29     0    0.00  14.50
2          1269     80.62  …  121.01       5    1059    29   36.52   8.81
3           241     84.56  …   76.32       0    1125    49   22.96   6.23
4            79     45.93  …  120.71      28       0     0    0.00   0.00
..          …       …  …     …     …     …   …     …    …
125           0      0.00  …  125.64       2       0     0    0.00   0.00
126        6814     75.78  …   42.85       0       0     0    0.00   0.00
127        8051     87.58  …  131.88      67     569    23   24.74   7.02
128         790     73.55  …   91.67       1    1783    65   27.43   7.75
129         343     95.81  …  122.22       0      99     2   49.50   9.00

     SR-BL  AUCTION YEAR  BASE PRICE  SOLD PRICE
0    13.93          2009       50000       50000
1     0.00          2008       50000       50000
2    24.90          2008      200000      350000
3    22.14          2011      100000      850000
4     0.00          2011      100000      800000
..     …           …         …         …
125   0.00          2010       50000      750000
126   0.00          2008      225000      225000
127  21.13          2011      400000     1800000
128  21.26          2008      200000      450000
129  33.00          2008      100000      110000

[130 rows x 26 columns]
```

[155]:
```
type(df)
```

[155]:
```
pandas.core.frame.DataFrame
```

[156]:
```
df.head(10)
```

[156]:
```
   Sl.NO.    PLAYER NAME  AGE COUNTRY  TEAM PLAYING ROLE  T-RUNS  T-WKTS  \
0       1    Abdulla, YA    2      SA  KXIP   Allrounder       0       0
1       2   Abdur Razzak    2     BAN   RCB       Bowler     214      18
2       3    Agarkar, AB    2     IND   KKR       Bowler     571      58
3       4      Ashwin, R    1     IND   CSK       Bowler     284      31
```

```
4       5   Badrinath, S    2     IND   CSK      Batsman      63       0
5       6     Bailey, GJ    2     AUS   CSK      Batsman       0       0
6       7      Balaji, L    2     IND  CSK+       Bowler      51      27
7       8  Bollinger, DE    2     AUS   CSK       Bowler      54      50
8       9       Botha, J    2      SA    RR   Allrounder      83      17
9      10    Boucher, MV    2      SA  RCB+    W. Keeper    5515       1

   ODI-RUNS-S  ODI-SR-B  …    SR-B  SIXERS  RUNS-C  WKTS  AVE-BL   ECON  \
0           0      0.00  …    0.00       0     307    15   20.47   8.90
1         657     71.41  …    0.00       0      29     0    0.00  14.50
2        1269     80.62  …  121.01       5    1059    29   36.52   8.81
3         241     84.56  …   76.32       0    1125    49   22.96   6.23
4          79     45.93  …  120.71      28       0     0    0.00   0.00
5         172     72.26  …   95.45       0       0     0    0.00   0.00
6         120     78.94  …   72.22       1    1342    52   25.81   7.98
7          50     92.59  …  165.88       1     693    37   18.73   7.22
8         609     85.77  …  114.73       3     610    19   32.11   6.85
9        4686     84.76  …  127.51      13       0     0    0.00   0.00

   SR-BL  AUCTION YEAR  BASE PRICE  SOLD PRICE
0  13.93          2009       50000       50000
1   0.00          2008       50000       50000
2  24.90          2008      200000      350000
3  22.14          2011      100000      850000
4   0.00          2011      100000      800000
5   0.00          2009       50000       50000
6  19.40          2011      100000      500000
7  15.57          2011      200000      700000
8  28.11          2011      200000      950000
9   0.00          2008      200000      450000

[10 rows x 26 columns]
```

[157]:
```
list(df)
```

[157]:
```
['Sl.NO.',
 'PLAYER NAME',
 'AGE',
 'COUNTRY',
 'TEAM',
 'PLAYING ROLE',
 'T-RUNS',
 'T-WKTS',
 'ODI-RUNS-S',
 'ODI-SR-B',
 'ODI-WKTS',
 'ODI-SR-BL',
```

```
 'CAPTAINCY EXP',
 'RUNS-S',
 'HS',
 'AVE',
 'SR-B',
 'SIXERS',
 'RUNS-C',
 'WKTS',
 'AVE-BL',
 'ECON',
 'SR-BL',
 'AUCTION YEAR',
 'BASE PRICE',
 'SOLD PRICE']
```

[159]:
```
df.head().transpose()
```

[159]:

	0	1	2	3	4
Sl.NO.	1	2	3	4	5
PLAYER NAME	Abdulla, YA	Abdur Razzak	Agarkar, AB	Ashwin, R	Badrinath, S
AGE	2	2	2	1	2
COUNTRY	SA	BAN	IND	IND	IND
TEAM	KXIP	RCB	KKR	CSK	CSK
PLAYING ROLE	Allrounder	Bowler	Bowler	Bowler	Batsman
T-RUNS	0	214	571	284	63
T-WKTS	0	18	58	31	0
ODI-RUNS-S	0	657	1269	241	79
ODI-SR-B	0.0	71.41	80.62	84.56	45.93
ODI-WKTS	0	185	288	51	0
ODI-SR-BL	0.0	37.6	32.9	36.8	0.0
CAPTAINCY EXP	0	0	0	0	0
RUNS-S	0	0	167	58	1317
HS	0	0	39	11	71
AVE	0.0	0.0	18.56	5.8	32.93
SR-B	0.0	0.0	121.01	76.32	120.71
SIXERS	0	0	5	0	28
RUNS-C	307	29	1059	1125	0
WKTS	15	0	29	49	0
AVE-BL	20.47	0.0	36.52	22.96	0.0
ECON	8.9	14.5	8.81	6.23	0.0
SR-BL	13.93	0.0	24.9	22.14	0.0
AUCTION YEAR	2009	2008	2008	2011	2011
BASE PRICE	50000	50000	200000	100000	100000
SOLD PRICE	50000	50000	350000	850000	800000

[160]:
```
df.shape
```

[160]: (130, 26)

[161]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 130 entries, 0 to 129
Data columns (total 26 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Sl.NO.         130 non-null    int64
 1   PLAYER NAME    130 non-null    object
 2   AGE            130 non-null    int64
 3   COUNTRY        130 non-null    object
 4   TEAM           130 non-null    object
 5   PLAYING ROLE   130 non-null    object
 6   T-RUNS         130 non-null    int64
 7   T-WKTS         130 non-null    int64
 8   ODI-RUNS-S     130 non-null    int64
 9   ODI-SR-B       130 non-null    float64
 10  ODI-WKTS       130 non-null    int64
 11  ODI-SR-BL      130 non-null    float64
 12  CAPTAINCY EXP  130 non-null    int64
 13  RUNS-S         130 non-null    int64
 14  HS             130 non-null    int64
 15  AVE            130 non-null    float64
 16  SR-B           130 non-null    float64
 17  SIXERS         130 non-null    int64
 18  RUNS-C         130 non-null    int64
 19  WKTS           130 non-null    int64
 20  AVE-BL         130 non-null    float64
 21  ECON           130 non-null    float64
 22  SR-BL          130 non-null    float64
 23  AUCTION YEAR   130 non-null    int64
 24  BASE PRICE     130 non-null    int64
 25  SOLD PRICE     130 non-null    int64
dtypes: float64(7), int64(15), object(4)
memory usage: 26.5+ KB
```

[162]:
```
df[0:5]
```

[162]:

	Sl.NO.	PLAYER NAME	AGE	COUNTRY	TEAM	PLAYING ROLE	T-RUNS	T-WKTS	\
0	1	Abdulla, YA	2	SA	KXIP	Allrounder	0	0	
1	2	Abdur Razzak	2	BAN	RCB	Bowler	214	18	
2	3	Agarkar, AB	2	IND	KKR	Bowler	571	58	
3	4	Ashwin, R	1	IND	CSK	Bowler	284	31	
4	5	Badrinath, S	2	IND	CSK	Batsman	63	0	

```
   ODI-RUNS-S  ODI-SR-B  …    SR-B  SIXERS  RUNS-C  WKTS  AVE-BL   ECON  \
0           0      0.00  …    0.00       0     307    15   20.47   8.90
1         657     71.41  …    0.00       0      29     0    0.00  14.50
2        1269     80.62  …  121.01       5    1059    29   36.52   8.81
3         241     84.56  …   76.32       0    1125    49   22.96   6.23
4          79     45.93  …  120.71      28       0     0    0.00   0.00

   SR-BL  AUCTION YEAR  BASE PRICE  SOLD PRICE
0  13.93          2009       50000       50000
1   0.00          2008       50000       50000
2  24.90          2008      200000      350000
3  22.14          2011      100000      850000
4   0.00          2011      100000      800000

[5 rows x 26 columns]
```

[163]:
```
df[:5]
```

[163]:
```
   Sl.NO.   PLAYER NAME  AGE COUNTRY  TEAM PLAYING ROLE  T-RUNS  T-WKTS  \
0       1   Abdulla, YA    2      SA  KXIP   Allrounder       0       0
1       2  Abdur Razzak    2     BAN   RCB       Bowler     214      18
2       3   Agarkar, AB    2     IND   KKR       Bowler     571      58
3       4     Ashwin, R    1     IND   CSK       Bowler     284      31
4       5  Badrinath, S    2     IND   CSK      Batsman      63       0

   ODI-RUNS-S  ODI-SR-B  …    SR-B  SIXERS  RUNS-C  WKTS  AVE-BL   ECON  \
0           0      0.00  …    0.00       0     307    15   20.47   8.90
1         657     71.41  …    0.00       0      29     0    0.00  14.50
2        1269     80.62  …  121.01       5    1059    29   36.52   8.81
3         241     84.56  …   76.32       0    1125    49   22.96   6.23
4          79     45.93  …  120.71      28       0     0    0.00   0.00

   SR-BL  AUCTION YEAR  BASE PRICE  SOLD PRICE
0  13.93          2009       50000       50000
1   0.00          2008       50000       50000
2  24.90          2008      200000      350000
3  22.14          2011      100000      850000
4   0.00          2011      100000      800000

[5 rows x 26 columns]
```

[164]:
```
df[-5:]
```

[164]:
```
     Sl.NO.    PLAYER NAME  AGE COUNTRY   TEAM PLAYING ROLE  T-RUNS  T-WKTS  \
125     126      Yadav, AS    2     IND     DC      Batsman       0       0
126     127   Younis Khan     2     PAK     RR      Batsman    6398       7
127     128  Yuvraj Singh     2     IND  KXIP+      Batsman    1775       9
```

```
128     129   Zaheer Khan    2     IND    MI+       Bowler    1114     288
129     130    Zoysa, DNT    2      SL     DC       Bowler     288      64

     ODI-RUNS-S  ODI-SR-B  …    SR-B  SIXERS  RUNS-C  WKTS  AVE-BL  ECON  \
125           0      0.00  …  125.64       2       0     0    0.00  0.00
126        6814     75.78  …   42.85       0       0     0    0.00  0.00
127        8051     87.58  …  131.88      67     569    23   24.74  7.02
128         790     73.55  …   91.67       1    1783    65   27.43  7.75
129         343     95.81  …  122.22       0      99     2   49.50  9.00

     SR-BL  AUCTION YEAR  BASE PRICE  SOLD PRICE
125   0.00          2010       50000      750000
126   0.00          2008      225000      225000
127  21.13          2011      400000     1800000
128  21.26          2008      200000      450000
129  33.00          2008      100000      110000

[5 rows x 26 columns]
```

[165]:
```
df[['PLAYER NAME','PLAYING ROLE']][:5]
```

[165]:
```
     PLAYER NAME PLAYING ROLE
0   Abdulla, YA   Allrounder
1  Abdur Razzak       Bowler
2   Agarkar, AB       Bowler
3     Ashwin, R       Bowler
4  Badrinath, S      Batsman
```

[166]:
```
df.iloc[4:9,1:4]
```

[166]:
```
      PLAYER NAME  AGE COUNTRY
4   Badrinath, S    2     IND
5     Bailey, GJ    2     AUS
6      Balaji, L    2     IND
7  Bollinger, DE    2     AUS
8       Botha, J    2      SA
```

[167]:
```
df.COUNTRY.value_counts()
```

[167]:
```
COUNTRY
IND    53
AUS    22
SA     16
SL     12
PAK     9
NZ      7
WI      6
```

```
ENG     3
BAN     1
ZIM     1
Name: count, dtype: int64
```

[168]:
```
df
```

[168]:
```
     Sl.NO.   PLAYER NAME  AGE COUNTRY   TEAM PLAYING ROLE  T-RUNS  T-WKTS  \
0         1   Abdulla, YA    2      SA   KXIP   Allrounder       0       0
1         2  Abdur Razzak    2     BAN    RCB       Bowler     214      18
2         3   Agarkar, AB    2     IND    KKR       Bowler     571      58
3         4     Ashwin, R    1     IND    CSK       Bowler     284      31
4         5  Badrinath, S    2     IND    CSK      Batsman      63       0
..      …           …  …     …    …           …      …     …
125     126     Yadav, AS    2     IND     DC      Batsman       0       0
126     127   Younis Khan    2     PAK     RR      Batsman    6398       7
127     128  Yuvraj Singh    2     IND  KXIP+      Batsman    1775       9
128     129   Zaheer Khan    2     IND    MI+       Bowler    1114     288
129     130    Zoysa, DNT    2      SL     DC       Bowler     288      64

     ODI-RUNS-S  ODI-SR-B  …    SR-B  SIXERS  RUNS-C  WKTS  AVE-BL   ECON  \
0             0      0.00  …    0.00       0     307    15   20.47   8.90
1           657     71.41  …    0.00       0      29     0    0.00  14.50
2          1269     80.62  …  121.01       5    1059    29   36.52   8.81
3           241     84.56  …   76.32       0    1125    49   22.96   6.23
4            79     45.93  …  120.71      28       0     0    0.00   0.00
..          …       …  …     …     …     …   …     …    …
125           0      0.00  …  125.64       2       0     0    0.00   0.00
126        6814     75.78  …   42.85       0       0     0    0.00   0.00
127        8051     87.58  …  131.88      67     569    23   24.74   7.02
128         790     73.55  …   91.67       1    1783    65   27.43   7.75
129         343     95.81  …  122.22       0      99     2   49.50   9.00

     SR-BL  AUCTION YEAR  BASE PRICE  SOLD PRICE
0    13.93          2009       50000       50000
1     0.00          2008       50000       50000
2    24.90          2008      200000      350000
3    22.14          2011      100000      850000
4     0.00          2011      100000      800000
..     …           …         …          …
125   0.00          2010       50000      750000
126   0.00          2008      225000      225000
127  21.13          2011      400000     1800000
128  21.26          2008      200000      450000
129  33.00          2008      100000      110000

[130 rows x 26 columns]
```

[169]:
```
df.AGE.value_counts()
```

[169]:
```
AGE
2    86
3    28
1    16
Name: count, dtype: int64
```

[170]:
```
df.COUNTRY.value_counts(normalize=True)*100
```

[170]:
```
COUNTRY
IND    40.769231
AUS    16.923077
SA     12.307692
SL      9.230769
PAK     6.923077
NZ      5.384615
WI      4.615385
ENG     2.307692
BAN     0.769231
ZIM     0.769231
Name: proportion, dtype: float64
```

[171]:
```
pd.crosstab(df['AGE'], df['PLAYING ROLE'])
```

[171]:

PLAYING ROLE	Allrounder	Batsman	Bowler	W. Keeper
AGE				
1	4	5	7	0
2	25	21	29	11
3	6	13	8	1

[172]:
```
df[['PLAYER NAME', 'SOLD PRICE']].sort_values('SOLD PRICE')
```

[172]:

	PLAYER NAME	SOLD PRICE
73	Noffke, AA	20000
46	Kamran Khan	24000
0	Abdulla, YA	50000
1	Abdur Razzak	50000
118	Van der Merwe	50000
..	…	…
113	Tiwary, SS	1600000
111	Tendulkar, SR	1800000
50	Kohli, V	1800000
93	Sehwag, V	1800000
127	Yuvraj Singh	1800000

[130 rows x 2 columns]

[173]:
```
df[['PLAYER NAME', 'SOLD PRICE']].sort_values('SOLD PRICE', ascending =False)
```

[173]:

	PLAYER NAME	SOLD PRICE
93	Sehwag, V	1800000
127	Yuvraj Singh	1800000
50	Kohli, V	1800000
111	Tendulkar, SR	1800000
113	Tiwary, SS	1600000
..	…	…
34	Henriques, MC	50000
5	Bailey, GJ	50000
0	Abdulla, YA	50000
46	Kamran Khan	24000
73	Noffke, AA	20000

[130 rows x 2 columns]

[175]:
```
df['PREMIUM'] = df['SOLD PRICE'] - df['BASE PRICE']
```

[176]:
```
df[['PLAYER NAME', 'BASE PRICE', 'SOLD PRICE']][0:5]
```

[176]:

	PLAYER NAME	BASE PRICE	SOLD PRICE
0	Abdulla, YA	50000	50000
1	Abdur Razzak	50000	50000
2	Agarkar, AB	200000	350000
3	Ashwin, R	100000	850000
4	Badrinath, S	100000	800000

[177]:
```
df[['PLAYER NAME', 'BASE PRICE', 'SOLD PRICE', 'PREMIUM']].
 ↪sort_values('PREMIUM', ascending =False)
```

[177]:

	PLAYER NAME	BASE PRICE	SOLD PRICE	PREMIUM
50	Kohli, V	150000	1800000	1650000
113	Tiwary, SS	100000	1600000	1500000
127	Yuvraj Singh	400000	1800000	1400000
111	Tendulkar, SR	400000	1800000	1400000
93	Sehwag, V	400000	1800000	1400000
..	…	…	…	…
102	Smith, DR	100000	100000	0
73	Noffke, AA	20000	20000	0
100	Silva, LPC	100000	100000	0
74	Ntini, M	200000	200000	0
0	Abdulla, YA	50000	50000	0

[130 rows x 4 columns]

[178]:
```
df.groupby('AGE')['SOLD PRICE'].mean()
```

[178]:
```
AGE
1    720250.000000
2    484534.883721
3    520178.571429
Name: SOLD PRICE, dtype: float64
```

[179]:
```
soldprice_by_age =df.groupby('AGE')['SOLD PRICE'].mean().reset_index()
print(soldprice_by_age)
```

```
   AGE     SOLD PRICE
0    1  720250.000000
1    2  484534.883721
2    3  520178.571429
```

[180]:
```
soldprice_by_age_role=df.groupby(['AGE', 'PLAYING ROLE'])['SOLD PRICE'].mean().
 ↪reset_index()
print(soldprice_by_age_role)
```

```
    AGE PLAYING ROLE    SOLD PRICE
0     1   Allrounder  5.875000e+05
1     1      Batsman  1.110000e+06
2     1       Bowler  5.177143e+05
3     2   Allrounder  4.494000e+05
4     2      Batsman  6.547619e+05
5     2       Bowler  3.979310e+05
6     2    W. Keeper  4.677273e+05
7     3   Allrounder  7.666667e+05
8     3      Batsman  4.576923e+05
9     3       Bowler  4.143750e+05
10    3    W. Keeper  7.000000e+05
```

[181]:
```
soldprice_comparison = soldprice_by_age_role.merge(soldprice_by_age, on =␣
 ↪'AGE',how= 'outer')
print(soldprice_comparison)
```

```
    AGE PLAYING ROLE  SOLD PRICE_x   SOLD PRICE_y
0     1   Allrounder  5.875000e+05  720250.000000
1     1      Batsman  1.110000e+06  720250.000000
2     1       Bowler  5.177143e+05  720250.000000
3     2   Allrounder  4.494000e+05  484534.883721
4     2      Batsman  6.547619e+05  484534.883721
5     2       Bowler  3.979310e+05  484534.883721
6     2    W. Keeper  4.677273e+05  484534.883721
7     3   Allrounder  7.666667e+05  520178.571429
8     3      Batsman  4.576923e+05  520178.571429
9     3       Bowler  4.143750e+05  520178.571429
10    3    W. Keeper  7.000000e+05  520178.571429
```

[182]:
```
soldprice_comparison.rename(columns ={'SOLD PRICE_x':␣
↪'SOLD_PRICE_AGE_ROLE','SOLD PRICE_y':'SOLD_PRICE_AGE' })
```

[182]:

	AGE	PLAYING ROLE	SOLD_PRICE_AGE_ROLE	SOLD_PRICE_AGE
0	1	Allrounder	5.875000e+05	720250.000000
1	1	Batsman	1.110000e+06	720250.000000
2	1	Bowler	5.177143e+05	720250.000000
3	2	Allrounder	4.494000e+05	484534.883721
4	2	Batsman	6.547619e+05	484534.883721
5	2	Bowler	3.979310e+05	484534.883721
6	2	W. Keeper	4.677273e+05	484534.883721
7	3	Allrounder	7.666667e+05	520178.571429
8	3	Batsman	4.576923e+05	520178.571429
9	3	Bowler	4.143750e+05	520178.571429
10	3	W. Keeper	7.000000e+05	520178.571429

[187]:
```
soldprice_comparison['change'] = soldprice_comparison.apply(lambda rec:(rec.
↪SOLD_PRICE_AGE_ROLE - rec.SOLD_PRICE_AGE) / rec.SOLD_PRICE_AGE, axis = 1)
!head
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_43201/363465696.py in ?()
----> 1 soldprice_comparison['change'] = soldprice_comparison.apply(lambda rec:
 ↪(rec.SOLD_PRICE_AGE_ROLE - rec.SOLD_PRICE_AGE) / rec.SOLD_PRICE_AGE, axis = 1
      2 get_ipython().system('head')

~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/frame.py in ?
 ↪(self, func, axis, raw, result_type, args, by_row, engine, engine_kwargs,␣
 ↪**kwargs)
  10370             engine_kwargs=engine_kwargs,
  10371             args=args,
  10372             kwargs=kwargs,
  10373         )
> 10374         return op.apply().__finalize__(self, method="apply")

~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/apply.py in ?
 ↪(self)
    912         # raw
    913         elif self.raw:
    914             return self.apply_raw(engine=self.engine, engine_kwargs=sel
 ↪engine_kwargs)
    915
--> 916         return self.apply_standard()

~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/apply.py in ?
 ↪(self)
```

```
   1061     def apply_standard(self):
   1062         if self.engine == "python":
-> 1063             results, res_index = self.apply_series_generator()
   1064         else:
   1065             results, res_index = self.apply_series_numba()
   1066

~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/apply.py in ?
 ↪(self)
   1077
   1078         with option_context("mode.chained_assignment", None):
   1079             for i, v in enumerate(series_gen):
   1080                 # ignore SettingWithCopy here in case the user mutates
-> 1081                 results[i] = self.func(v, *self.args, **self.kwargs)
   1082                 if isinstance(results[i], ABCSeries):
   1083                     # If we have a view on v, we need to make a copy␣
 ↪because
   1084                     #  series_generator will swap out the underlying da a

/tmp/ipykernel_43201/363465696.py in ?(rec)
----> 1 soldprice_comparison['change'] = soldprice_comparison.apply(lambda rec:
 ↪(rec.SOLD_PRICE_AGE_ROLE - rec.SOLD_PRICE_AGE) / rec.SOLD_PRICE_AGE, axis = 1

~/anaconda3/envs/pandas/lib/python3.12/site-packages/pandas/core/generic.py in
 ↪(self, name)
   6295             and name not in self._accessors
   6296             and self._info_axis.
 ↪_can_hold_identifiers_and_holds_name(name)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)

AttributeError: 'Series' object has no attribute 'SOLD_PRICE_AGE_ROLE'
```

[184]:
```
df[df['SIXERS']>80] [['PLAYER NAME', 'SIXERS']]
```

[184]:
```
      PLAYER NAME  SIXERS
26      Gayle, CH     129
28  Gilchrist, AC      86
82     Pathan, YK      81
88      Raina, SK      97
97     Sharma, RG      82
```

[185]:
```
df
```

[185]:
```
     Sl.NO.   PLAYER NAME  AGE COUNTRY   TEAM PLAYING ROLE  T-RUNS  T-WKTS  \
0         1   Abdulla, YA    2      SA   KXIP   Allrounder       0       0
```

```
1          2  Abdur Razzak    2     BAN    RCB       Bowler     214       18
2          3   Agarkar, AB    2     IND    KKR       Bowler     571       58
3          4     Ashwin, R    1     IND    CSK       Bowler     284       31
4          5  Badrinath, S    2     IND    CSK      Batsman      63        0
..       ...           ...  ...     ...    ...          ...     ...      ...
125      126     Yadav, AS    2     IND     DC      Batsman       0        0
126      127   Younis Khan    2     PAK     RR      Batsman    6398        7
127      128  Yuvraj Singh    2     IND  KXIP+      Batsman    1775        9
128      129   Zaheer Khan    2     IND    MI+       Bowler    1114      288
129      130    Zoysa, DNT    2      SL     DC       Bowler     288       64

     ODI-RUNS-S  ODI-SR-B  …  SIXERS  RUNS-C  WKTS  AVE-BL   ECON  SR-BL  \
0             0      0.00  …       0     307    15   20.47   8.90  13.93
1           657     71.41  …       0      29     0    0.00  14.50   0.00
2          1269     80.62  …       5    1059    29   36.52   8.81  24.90
3           241     84.56  …       0    1125    49   22.96   6.23  22.14
4            79     45.93  …      28       0     0    0.00   0.00   0.00
..          ...       ...  ...   ...     ...   ...     ...    ...    ...
125           0      0.00  …       2       0     0    0.00   0.00   0.00
126        6814     75.78  …       0       0     0    0.00   0.00   0.00
127        8051     87.58  …      67     569    23   24.74   7.02  21.13
128         790     73.55  …       1    1783    65   27.43   7.75  21.26
129         343     95.81  …       0      99     2   49.50   9.00  33.00

     AUCTION YEAR  BASE PRICE  SOLD PRICE  PREMIUM
0            2009       50000       50000        0
1            2008       50000       50000        0
2            2008      200000      350000   150000
3            2011      100000      850000   750000
4            2011      100000      800000   700000
..            ...         ...         ...      ...
125          2010       50000      750000   700000
126          2008      225000      225000        0
127          2011      400000     1800000  1400000
128          2008      200000      450000   250000
129          2008      100000      110000    10000

[130 rows x 27 columns]
```

[186]:
```
autos = pd.read_csv('auto-mpg.data', sep = '\s+', header = None)
```

```
<>:1: SyntaxWarning: invalid escape sequence '\s'
<>:1: SyntaxWarning: invalid escape sequence '\s'
/tmp/ipykernel_43201/62692210.py:1: SyntaxWarning: invalid escape sequence '\s'
  autos = pd.read_csv('auto-mpg.data', sep = '\s+', header = None)
```

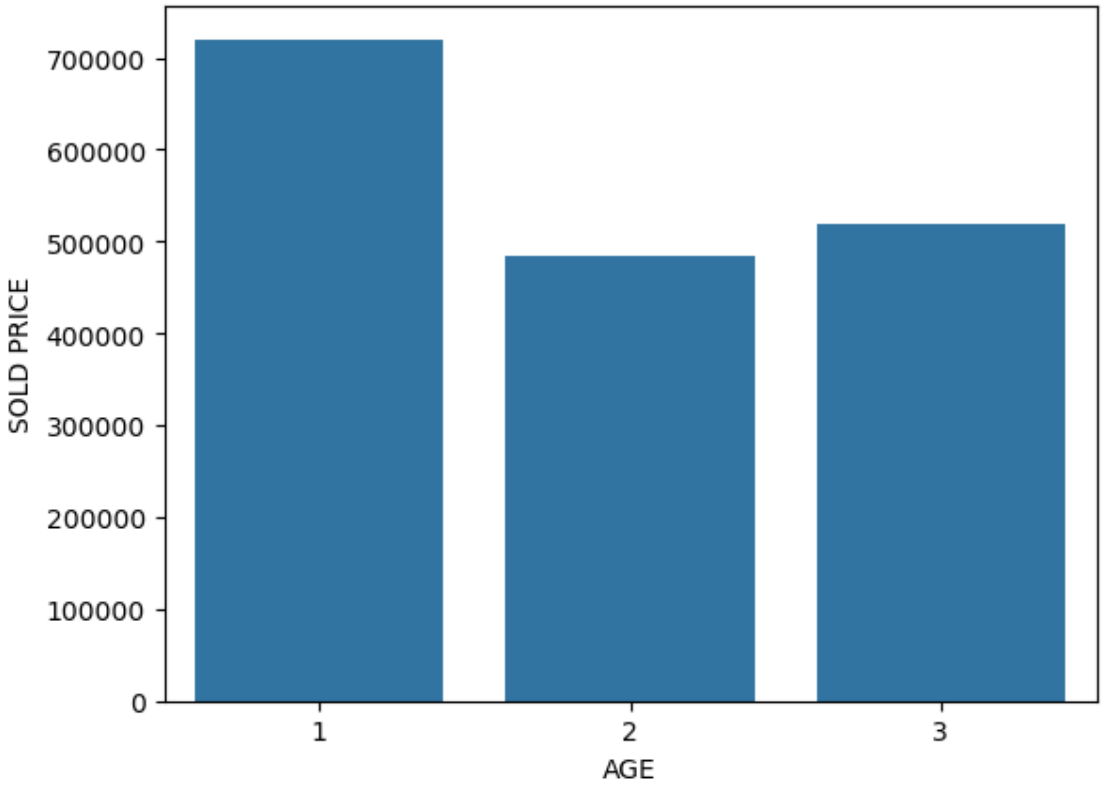
7 Visualization

[189]:
```
!pip install seaborn
import matplotlib.pyplot as plt
import seaborn as sn
# %matplotlib inline
```

```
Requirement already satisfied: seaborn in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from seaborn)
(1.26.4)
Requirement already satisfied: pandas>=1.2 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from seaborn)
(2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from seaborn)
(3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=8 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from
pandas>=1.2->seaborn) (2023.3)
```

```
Requirement already satisfied: six>=1.5 in
/home/test/anaconda3/envs/pandas/lib/python3.12/site-packages (from python-
dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```

[190]:
```
sn.barplot(x='AGE', y = 'SOLD PRICE', data = soldprice_by_age);
```



7.0.1 Bar Chart

Bar chart is a frequency chart for qualitative variable (or categorical variable). Bar chart can be used to assess the most-occurring and least-occurring categories within a dataset.

7.0.2 Histogram

A histogram is a plot that shows the frequency distribution of a set of continuous variable. Histogram gives an insight into the underlying distribution (e.g., normal distribution) of the variable, outliers, skew- ness, etc

7.0.3 Distribution or Density Plot

A distribution or density plot depicts the distribution of data over a continuous interval. Density plot is like smoothed histogram and visualizes distribution of data over a continuous interval. So, a density plot also gives insight into what might be the distribution of the population.

7.0.4 Box Plot

Box plot (aka Box and Whisker plot) is a graphical representation of numerical data that can be used to understand the variability of the data and the existence of outliers. Box plot is designed by identifying the following descriptive statistics: 1. Lower quartile (1st quartile), median and upper quartile (3rd quartile). 2. Lowest and highest values. 3. Inter-quartile range (IQR).

7.0.5 Comparing Distributions

The distribution for different categories can be compared by overlapping the distributions.

7.0.6 Scatter Plot

In a scatter plot, the values of two variables are plotted along two axes and resulting pattern can reveal cor- relation present between the variables, if any. The relationship could be linear or non-linear. A scatter plot is also useful for assessing the strength of the relationship and to find if there are any outliers in the data. Scatter plots are used during regression model building to decide on the initial model, that is whether to include a variable in a regression model or not.

7.0.7 Pair Plot

If there are many variables, it is not convenient to draw scatter plots for each pair of variables to under- stand the relationships. So, a pair plot can be used to depict the relationships in a single diagram which can be plotted using pairplot() method

7.0.8 Correlation and Heatmap

Correlation is used for measuring the strength and direction of the linear relationship between two continuous random variables X and Y. It is a statistical measure that indicates the extent to which two variables change together. A positive correlation means the variables increase or decrease together; a negative correlation means if one variable increases, the other decreases.